Concurrent processes as wireless proof nets

Emmanuel Beffara[1] and Virgile Mogbil[2]

[1] IML - UMR6206, CNRS - Université Aix-Marseille 2
[2] LIPN - UMR7030, CNRS - Université Paris 13

Abstract. We present a proofs-as-programs correspondence between linear logic and process calculi that permits non-deterministic behaviours. Processes are translated into wireless proof nets, i.e. proof nets of multiplicative linear logic without cut wires. Typing a term using them consists in typing some of its possible determinisations in standard sequent calculus. Generalized cut-elimination steps in wireless proof nets is shown to correspond to executions that avoid deadlocks.

1 Introduction

Proof nets [5] were introduced with linear logic as its natural proof syntax, and the graphical formulation of cut elimination in them emphasizes the fact that linear logic is a logic of interaction. However, twenty years later, the search for formal relationships with the theory of concurrency, in the form of a satisfactory proofs-as-programs correspondence, is still an active research domain.

A type system for processes was proposed by Yoshida, Berger and Honda [12] to characterize normalizable processes, and further study revealed thas it was very close to LL [6]. LL itself was used as a type system by the first author [1] with comparable results, providing an interpretation of proofs as concurrent processes. Nevertheless, these approaches are not satisfactory yet as a logical account of concurrency, because they lack the inherent non-determinism of actual concurrency, precisely because they use logic to gain determinism.

Several approaches to the question of non-determinism in logic were also proposed. The use of the additive connectives of LL as a proof-theoretic representation of it was for instance explored by Mairson and Terui [8] to provide a notion of non-determinsitic cut-elimination, or by Maurel [9] who used it to represent the kind of non-determinism familiar in complexity theory. In a different style, differential logic was recently developped by Ehrhard and Regnier [4] and its untyped proof formalism was shown to represent the π-calculus faithfully [3].

The present work proposes a new approach to combine these two independent treatments of concurrency in logic. We start from the well studied framework of proof-nets, and we gain expressiveness by relaxing the determinism constraints imposed by the discipline of cut wires in LL proofs. We obtain *wireless* proof-nets, a very liberal syntax for proofs which enjoys parallel and non-determistic reduction while preserving logical correctness using an extension of familiar criteria. Terms of a variant of the π-calculus are translated naturally in this formalism,

which gives a notion of typability that ensures a good behaviour without imposing determinism. The idea is that typability of a term in our sense means the existence of possibly different typings for different executions of this term. The formalisation of this notion reveals a fine-grained correspondance between operational aspects of both formalisms.

The correspondence we establish extends previous work on correspondences between logic and concurrent programs, and can serve as a starting point for various further studies. Wireless proof nets allow for the study of parallel computation in a logical setting, as shown by the second author in related work [10], and work is in progress on relationships with parallel complexity in the style of implicit comptuational complexity (work by Terui [11] in this direction is a notable source of inspiration in this respect). Parallel semantics for processes is another direction for future work, for instance it can be expected that type disciplines would allow for a study of parallel computation in a more explicit way than in usual semantics of process calculi.

The paper is organised as follows: In section 2, a form of π-calculus is defined, together with new tools for the study of its operational semantics. Section 3 defines an interfacing discipline and a notion of determinisation, and section 4 adapts the LL typing of processes [1] and recalls the properties of typed terms. Section 5 presents wireless proof nets and their cut-elimination, and section 6 establishes the correspondence between these two formalisms.

2 A polarized process calculus

Let $\mathcal{N}$ be an infinite set of *names*, partitioned into positive ($a, b, c \in \mathcal{N}^+$) and negative ($\bar{a}, \bar{b}, \bar{c} \in \mathcal{N}^-$) ones, with in involution (written $\bar{\cdot}$) between them. Names of arbitrary polarity are ranged over by letters x, y, z. The polarity of a name x is written $\operatorname{pol} x$, an element of $\{\downarrow, \uparrow\}$ for positive and negative respectively. Finite sequences of names are written as Greek letters. Let $\mathcal{L}$ be an arbitrary set of *locations*, used to identify occurrences of names without ambiguity.

Definition 1 (terms). *The set $\overrightarrow{\pi}$ of terms is defined by the following grammar with the restriction that all locations in any term are always pairwise distinct:*

$$\begin{array}{lll} P, Q ::= & x^\ell(\zeta).P & \textit{action, with } \ell \in \mathcal{L},\ x \in \mathcal{N},\ \zeta \in \mathcal{N}^* \\ & a^\ell \multimap \bar{b}^m & \textit{forwarder, with } \ell, m \in \mathcal{L},\ a \in \mathcal{N}^+,\ \bar{b} \in \mathcal{N}^- \\ & P \mid Q,\ 1,\ (\boldsymbol{\nu} x)P & \textit{parallel composition, inaction, hiding} \end{array}$$

By convention, inputs are actions on positive names and outputs are actions on negative ones. All actions communicate private names, i.e. $z_1 \dots z_n$ are bound in $x(z_1 \dots z_n).P$. Intuitively, $(\boldsymbol{\nu} x)$ closes a channel by connecting its input and output parts and making them private; hence $(\boldsymbol{\nu} x)$ is a binder for both x and $\bar{x}$, and it is equivalent to $(\boldsymbol{\nu} \bar{x})$. A name x is called *hidden* if it is bound by a $(\boldsymbol{\nu} x)$ and *action-bound* if it is bound by an action prefix. The set $FV(P)$ of free names

of P contains all other names. Terms are considered up to injective renaming of bound names, provided that renaming preserves the polarity of all names:

$$\begin{aligned}(\boldsymbol{\nu} a)P &= (\boldsymbol{\nu} b)P[b/a, \bar{b}/\bar{a}] \quad && \text{if } b, \bar{b} \notin FV(P)\\ x^{\ell}(\alpha y \beta).P &= x^{\ell}(\alpha z \beta).P[z/y] \quad && \text{if } \operatorname{pol} y = \operatorname{pol} z \text{ and } z, \bar{z} \notin FV(P) \cup \alpha \cup \beta\end{aligned}$$

Using renaming, we use the convention that in a term $x^{\ell}(z_1 \dots z_n).P$, the dual names $\bar{z}_1 \dots \bar{z}_n$ do not occur free in P.

Let $\mathcal{L}(P)$ be the set of locations occurring in P. Given $\ell \in \mathcal{L}(P)$, the *subject* of ℓ is the name tagged by ℓ, written $\operatorname{subj}_P \ell$, its *polarity* is that of its subject, written $\operatorname{pol}_P \ell$. If ℓ is tags an action $x(z_1 \dots z_k)$, then for each i, the name z_i is the *i-th object* of ℓ, written $\operatorname{obj}^i_P \ell$. For each free or bound name x, let $\sharp(P, x)$ be the number of occurrences of x in P, not including any binder. *Prefixing* is the partial order $\leq_P$ over $\mathcal{L}(P)$ such that $\ell <_P m$ for all subterm $x^{\ell}(\zeta).Q$ of P and all $m \in \mathcal{L}(Q)$. In each case, the subscript P can be omitted if it is clear.

Definition 2 (structural congruence). *Structural congruence is defined in a standard way so that parallel composition is associative and commutative and that hiding enjoys the usual scoping rules, adapted to the polarity discipline:*

$$\begin{aligned}P \mid (\boldsymbol{\nu} a)Q &\equiv (\boldsymbol{\nu} a)(P \mid Q) && \textit{scope extrusion, if } a, \bar{a} \notin FV(P),\\ (\boldsymbol{\nu} a)P &\equiv P && \textit{garbage collection, if } a, \bar{a} \notin FV(P).\end{aligned}$$

Definition 3 (pairing). *A* pairing *of a term P is a partial involution c over $\mathcal{L}(P)$ such that for all $\ell \in \operatorname{dom} c$, $\operatorname{pol} c(\ell) = \overline{\operatorname{pol} \ell}$, and either $\operatorname{subj} c(\ell) = \overline{\operatorname{subj} \ell}$, or $\operatorname{subj} \ell = \operatorname{obj}^i m$ and $\operatorname{subj} c(\ell) = \operatorname{obj}^i c(m)$ for some $m \in \mathcal{L}(P)$ and $i \in \mathbb{N}$.*

For a pairing c, let $\sim_c$ be the smallest equivalence that contains c and all the (ℓ, m) such that there is a forwarder $a^{\ell} \multimap \bar{b}^m$. The pairing c is consistent *if $\operatorname{dom} c$ is closed for $\sim_c$, downwards closed for $\leq_P$ and $\leq_P$ quotiented by $\sim_c$ is acyclic.*

Observe that pairings and consistency are preserved by structural congruence, as a direct consequence of the fact that subjects, objects, polarities and prefixing are preserved by all structural congruence rules.

Definition 4 (execution). *Execution is the relation over terms up to structural congruence, labelled by involutions over $\mathcal{L}$, defined by the following rule in any context made of parallel compositions and hidings, for any $n \geq 0$:*

$$\begin{aligned}(\boldsymbol{\nu} a_0) \dots (\boldsymbol{\nu} a_n)\Big(\bar{a}_0^{\ell_0}(\bar{\zeta}).P \mid a_0^{m_0} \multimap \bar{a}_1^{\ell_1} \mid \dots \mid a_{n-1}^{m_{n-1}} \multimap \bar{a}_n^{\ell_n} \mid a_n^{m_n}(\zeta).Q \mid R\Big)&\\ \to_{ex}^{\{(\ell_i, m_i)\}_{i=0}^n} (\boldsymbol{\nu} a_0) \dots (\boldsymbol{\nu} a_n)\big((\boldsymbol{\nu}\zeta)(P \mid Q) \mid R\big)&\end{aligned}$$

Note that this rule assumes that the objects in the synchronizing actions have dual names. As a consequence, actions can synchronize only if their objets have the same arity and opposite polarities. The annotation c in $P \to_{ex}^{c} Q$ describes which occurrences are matched in the execution step, we write $P \to_{ex} Q$ if c is unimportant. Similarly, we keep locations implicit when they do not matter.

Remark that for two terms P and Q, several pairings c may correspond to a reduction $P \to_{ex}^{c} Q$, because of the role of forwarders. For instance, the execution

$$(\boldsymbol{\nu} u)\big(\bar{u}^a(x).P \mid u^b \multimap \bar{u}^c \mid u^d \multimap \bar{u}^e \mid u^f(\bar{x}).Q\big) \to_{ex} (\boldsymbol{\nu} ux)(P \mid Q)$$

can be annotated either $\{(a,b),(c,d),(e,f)\}$ or $\{(a,d),(e,b),(c,f)\}$. However, for a given c, there is at most one term Q such that $P \to_{ex}^{c} Q$, since the pairing c contains enough information to identify which subterms of P interact.

Lemma 1. *Let $P \to_{ex}^{c} Q$ be an execution and d be a pairing of Q, then* $\operatorname{dom} c \cap \operatorname{dom} d = \emptyset$ *and $c \cup d$ is a pairing of P. If d is consistent, then so is $c \cup d$.*

By iterating this, from an execution $P_0 \to_{ex}^{c_1} P_1 \to_{ex}^{c_2} \cdots \to_{ex}^{c_n} P_n$ we can deduce a pairing $c = c_1 \cup \cdots \cup c_n$ of P_0. This pairing represents the execution above, because it contains all the choices made during this execution. Indeed we can prove that executions that yield the same pairing are equivalent.

Lemma 2. *Let $P \to_{ex}^{c_1} Q_1$ and $P \to_{ex}^{c_2} Q_2$ be two executions with* $\operatorname{dom} c_1 \cap \operatorname{dom} c_2 = \emptyset$. *Then there is a unique R such that $Q_1 \to_{ex}^{c_2} R$ and $Q_2 \to_{ex}^{c_1} R$.*

As an interesting consequence of this fact, if in two execution sequences the union of the annotations are the same, then the sequences are necessarily a permutation of each other, and the final terms are the same. This supports the idea that pairings represent executions, forgetting inessential scheduling decisions.

Proposition 1. *A pairing c of a term P_0 is consistent if and only if there is an execution sequence $P_0 \to_{ex}^{c_1} \cdots \to_{ex}^{c_n} P_n$ such that $c = \bigcup_{i=1}^{n} c_i$.*

Proof (sketch). Remark that the set $\{\operatorname{dom} c_i\}$ must be the set of equivalence classes for $\sim_c$ in $\operatorname{dom} c$. Consistency ensures that $\leq_P$ is an order over these classes, and we get an run of P_0 by picking any total extension of this order.

Definition 5 (liveness). *A term P exhibits a forward $a \multimap \bar{b}$ if there is an execution step $(\boldsymbol{\nu} ab)(P \mid \bar{a}^{\ell} \mid b^m) \to_{ex}^{c} P'$ with $\ell, m \in \operatorname{dom} c$. P exhibits an action on a if there is an execution step $(\boldsymbol{\nu} a)(P \mid \bar{a}^{\ell}(\zeta)) \to_{ex}^{c} P'$ with $\ell \in \operatorname{dom} c$ for some ζ. P is* alive *if it has a reduct (by any number of execution steps including 0) that exhibits a forward or an action.*

3 Interfacing and determinisation

Definition 6. *Assume an infinite set $\mathfrak{V}$ of* indeterminates *ranged over by α. Sorts are defined as follows:*

name sorts	$\iota := \epsilon I,\ \alpha,\ \bar{\alpha}$	*with $\epsilon \in \{\downarrow, \uparrow\}$ and $\alpha \in \mathfrak{V}$,*
channel sorts	$I := n_1\iota_1, \ldots, n_k\iota_k$	*with $k, n_i \geq 0$,*
interfaces	$\mathcal{I} := x_1 : n_1\iota_1, \ldots, x_k : n_k\iota_k$	*with $k, n_i \geq 0$.*

The rules for interfacing of processes are given in table 1.

$$\frac{P :: \mathcal{I}, x : n\epsilon(n_1\iota_1, \ldots, n_k\iota_k), \{z_i : n_i\iota_i\}_{i=1}^k \quad \epsilon = \mathrm{pol}\, x}{x(z_1 \ldots z_n).P :: \mathcal{I}, x : (n+1)\epsilon(n_1\iota_1, \ldots, n_k\iota_k)} \qquad \frac{\iota \text{ is positive}}{a \multimap \bar{b} :: a : 1\iota, \bar{b} : 1\bar{\iota}}$$

$$\frac{P :: \{x_i : p_i\iota_i\}_{i=1}^k \quad Q :: \{x_i : q_i\iota_i\}_{i=1}^k}{P \mid Q :: \{x_i : (p_i + q_i)\iota_i\}_{i=1}^k} \qquad \frac{}{1 :: \emptyset} \qquad \frac{P :: \mathcal{I}, a : n{\downarrow}I, \bar{a} : n{\uparrow}\bar{I}}{(\boldsymbol{\nu} a)P :: \mathcal{I}}$$

Table 1. Interfacing rules

Positive name sorts are $\downarrow I$ and α, negative ones are $\uparrow I$ and $\bar{\alpha}$. The k in a channel sort is the *arity* and the n_i are the *multiplicities*. In the present work, we assume that the sum of multiplicities in any channel sort is never zero[3] The x_i in interfaces are pairwise distinct and interfaces are considered up to permutation and addition of nullary assignements. The *dual* $\bar{\iota}$ of a name sort ι is obtained by inverting all polarities in ι and replacing each occurrence of an indeterminate α by $\bar{\alpha}$, and vice-versa. The definition extends to channel sorts and interfaces.

Definition 7 (determinism). *A term is* deterministic *if it has an interface derivation where channel sorts only use multiplicity* 1 *and interfaces only use multiplicities* 1 *and* 0*.*

It is easily checked that interfacing is preserved by structural congruence and execution, since it is designed for this purpose. Besides, the very strong constraint on deterministic terms guarantees a very regular behaviour:

Proposition 2. *Every deterministic term has a unique maximal pairing. Execution in deterministic terms is strongly confluent.*

We now describe a way to turn arbitrary interfaced terms into deterministic terms. Interfaces are strong contraints on processes, mainly because of multiplicities, and the only actual source of non-determinism is how actions on a given channel are paired. A prototypical example is the term

$$T = (\boldsymbol{\nu} a)\big(\bar{a}(x).P_1 \mid \bar{a}(x).P_2 \mid a(\bar{x}).Q_1 \mid a(\bar{x}).Q_2\big)$$

which has two cases of execution (and two schedulings for each case), one leading to $(\boldsymbol{\nu} x)(P_1 \mid Q_1) \mid (\boldsymbol{\nu} x)(P_2 \mid Q_2)$ and the other to $(\boldsymbol{\nu} x)(P_1 \mid Q_2) \mid (\boldsymbol{\nu} x)(P_2 \mid Q_1)$. Of course this situation can occur indirectly, after other communications.

Hence a way to make a process deterministic is to associate each action with the action it will eventually synchronise with. Pairings describe this information, and the idea to impose a pairing is to split each name a into multiple names $a^1, \ldots, a^k$ with one occurrence each, so that a^i will only synchronize with $\bar{a}^i$. Moreover, we have to do that also for hidden and action-bound names, according to some pairing. To implement this, we assume an injective function that associates to each name x and integer i a name written x^i (such a function obviously exists since the set of names is infinite).

[3] This is just for simplifying the subsequent theory of proof nets, and it is not a strong constraint with respect to expressiveness.

Definition 8 (numbering). *A* numbering *is a function from $\mathcal{L}$ to $\mathbb{N}$. For a term P, a* numbering of P *is a numbering ϕ such that*

- *for all $\ell \in \mathcal{L}(P)$, $1 \leq \phi(\ell) \leq \sharp(P, \operatorname{subj} \ell)$,*
- *the mapping $\ell \mapsto (\operatorname{subj} \ell, \phi(\ell))$ is injective.*

Given a numbering ϕ, the determinisation *of P according to ϕ is the term $\lfloor P \rfloor_\phi$ defined inductively as follows:*

$$\lfloor x^\ell(z_1 \dots z_k).P \rfloor_\phi := x^{\phi(\ell)}(z_1^1 \dots z_1^{\sharp(P,z_1)} \dots z_k^1 \dots z_k^{\sharp(P,z_k)}).\, \lfloor P \rfloor_\phi$$

$$\lfloor (\boldsymbol{\nu} a) P \rfloor_\phi := (\boldsymbol{\nu} a^1 \dots a^{\max(\sharp(P,a),\sharp(P,\bar{a}))})\, \lfloor P \rfloor_\phi$$

$$\lfloor a^\ell \multimap \bar{b}^m \rfloor_\phi := a^{\phi(\ell)} \multimap \bar{b}^{\phi(m)} \qquad \lfloor P \mid Q \rfloor_\phi := \lfloor P \rfloor_\phi \mid \lfloor Q \rfloor_\phi \qquad \lfloor 1 \rfloor_\phi := 1$$

Proposition 3. *Let the* determinisation *of an interface $\mathcal{I}$ be the interface $\lfloor \mathcal{I} \rfloor$ defined inductively as follows, where I^n denotes n successive occurrences of I:*

$$\lfloor \{x_i : n_i \iota_i\}_{1 \leq i \leq k} \rfloor := \{x_i^j : 1 \lfloor \iota_i \rfloor\}_{\substack{1 \leq i \leq k \\ 1 \leq j \leq n_i}}$$

$$\lfloor \epsilon(n_1 \iota_1 \dots n_k \iota_k) \rfloor := \epsilon((1 \lfloor \iota_1 \rfloor)^{n_1} \dots (1 \lfloor \iota_k \rfloor)^{n_k})$$

For all term $P :: \mathcal{I}$ and numbering ϕ of P, $\lfloor P \rfloor_\phi$ is deterministic and $\lfloor P \rfloor_\phi :: \lfloor \mathcal{I} \rfloor$.

As a consequence of propositions 3 and 2, for any interfaced term P and any numbering ϕ of P, the determinisation $\lfloor P \rfloor_\phi$ has a unique maximal pairing c_ϕ. The following proposition is a converse of this fact.

Proposition 4. *For all interfaced term P and maximal pairing c of P, there is numbering ϕ of P such that c is the maximal pairing of $\lfloor P \rfloor_\phi$.*

4 Linear Type System

We now define a typing relation between terms and formulas of a presentation of multiplicative linear logic with unlimited arities, called $\mathrm{MLL_u}$.

Definition 9 (formulas). *The formulas (written $A, B, C, \dots$) of* $\mathrm{MLL_u}$ *are built from literals $\alpha, \alpha^\perp$ by n-ary multiplicative conjunction $\bigotimes_{i=1}^n A_i$ and disjunction $\parr_{i=1}^n A_i$, with $n > 1$ in each case. The* negation *$A^\perp$ of a non-literal formula A is defined by de Morgan duality as $(\bigotimes_{i=1}^n A_i)^\perp = \parr_{i=1}^n A_i^\perp$.*

A type *(written $\Gamma, \Delta \dots$) is a finite set of assignements $x_1 : A_1, \dots, x_n : A_n$ where the x_i are pairwise distinct names and the A_i are* $\mathrm{MLL_u}$ *formulas. Type derivations are built made from the rules of table 2. A term P has type Γ if there is a type derivation with conclusion $P \vdash \Gamma$.*

Formulas of $\mathrm{MLL_u}$ and deterministic name sorts are isomorphic in a direct way. Assuming that positive literals and indeterminate sorts are taken from the same set $\mathfrak{V}$, for a name sort ι, we define the formula $[\![\iota]\!]$ by induction as follows:

$$[\![\alpha]\!] = \alpha, \quad [\![\bar{\alpha}]\!] = \alpha^\perp, \quad [\![\downarrow(1\iota_1, \dots 1\iota_k)]\!] = \bigotimes_{i=1}^k [\![\iota_i]\!], \quad [\![\uparrow(1\iota_1, \dots 1\iota_k)]\!] = \parr_{i=1}^k [\![\iota_i]\!].$$

Axiom and cut (A is a literal α or a $\otimes$ formula):

$$\frac{}{a{-\!\circ}\bar{b} \vdash a : A, \bar{b} : A^{\perp}}\ (\text{ax}) \qquad \frac{P \vdash \Gamma, a : A \qquad Q \vdash \Delta, \bar{a} : A^{\perp}}{(\boldsymbol{\nu} a)(P \mid Q) \vdash \Gamma, \Delta}\ (\text{cut})$$

Multiplicatives (ζ stands for $z_1 \dots z_n$):

$$\frac{P \vdash \Gamma, z_1 : A_1, \dots, z_k : A_k}{\bar{a}(\zeta).P \vdash \Gamma, \bar{a} : \parr_{i=1}^{k} A_i}\ (\parr) \qquad \frac{\left\{P_i \vdash \Gamma_i, z_i : A_i\right\}_{i=1}^{k}}{a(\zeta).\prod_{i=1}^{k} P_i \vdash (\Gamma_i)_{i=1}^{k}, a : \bigotimes_{i=1}^{k} A_i}\ (\otimes)$$

Table 2. Inference rules in $\mathrm{MLL_u}$

Proposition 5. *For all term P and deterministic interface $\mathcal{I}$, if $P \vdash [\![\mathcal{I}]\!]$ then $P :: \mathcal{I}$. If $P :: \mathcal{I}$ and P is typable, then P is deterministic and $P \vdash [\![\mathcal{I}]\!]$.*

Hence typed terms are always deterministic. In the following statements we will thus identify locations and names, since in deterministic terms, each name has at most one occurrence, so for any name u there is at most one location ℓ such that $\mathrm{subj}\,\ell = u$. A type derivation Π for a term P corresponds uniquely to an assignement of types to names/locations in P, although obviously not all type assignements correspond to a type derivation. Given a derivation Π, we will thus denote by $\Pi(\ell)$ the $\mathrm{MLL_u}$ formula associated with the location ℓ in Π.

Definition 10 (typing constraint). *A typing constraint for a term P is an equivalence relation over $\mathcal{L}(P)$. A type derivation Π for P is said to respect the constraint R if for all locations $(\ell, m) \in R$, either $\mathrm{pol}_P\,\ell = \mathrm{pol}_P\,m$ and $\Pi(\ell) = \Pi(m)$, or $\mathrm{pol}_P\,\ell = \neg\,\mathrm{pol}_P\,m$ and $\Pi(\ell) = \Pi(m)^{\perp}$. If there is a derivation Π that respects R and with conclusion Γ, we write $P \vdash_R \Gamma$.*

Typing constraints impose that distinct names in a given term have the same type even if they are unrelated. The idea is to represent the fact that distinct names are actually occurrences of the same name after determinisation.

We consider type substitution as usual in the following:

Proposition 6. *Let P be a term and R a typing constraint for P. Let $(\Pi : P \vdash \Gamma_i)_{i \in I}$ be a family of type derivations of P. Then there exists a type Γ and a family of substitutions $(\sigma_i)_{i \in I}$ such that $P \vdash_R \Gamma$ and $\Gamma_i = \Gamma\sigma_i$ for each $i \in I$.*

Proof (sketch). Remark that R is just a unification constraint, and that typing rules simply impose further similar constraints. We then prove an extension of this result to a family of terms under a global constraint, by a standard unification technique, and the typing hypothesis guarantees the success of unification.

Corollary 1 (principal typing). *If a term P is typable under a constraint R, then there is a type Γ, unique up to renaming of the literals, such that all possible types of P under R are substitutions of Γ.*

The final property we prove for our type system is subject reduction. For this purpose, we define an annotated cut elimination relation $\rightarrow^c_{sc}$, where c is a pairing as of definition 3. Cut elimination steps are standard from the point of view of sequent calculus, only refined to handle locations. We describe them informally, according to the nature of the premisses of a given cut rule:

logical steps for multiplicatives. If the premisses are multiplicative introductions for the cut formulas, then the cut and these rules are replaced by one cut for each immediate subformula.

logical steps for axioms. If one premiss is an axiom rule for the cut formula, then this rule and the cut are removed, and in the other premiss the location ℓ of the cut formula is replaced by that of the other side of the axiom:

$$\dfrac{\Pi : P \vdash \Gamma, \bar{a} : A^\perp \qquad \dfrac{}{a^m \multimap \bar{b}^n \vdash a : A, \bar{b} : A^\perp}\,(\text{ax})}{(\boldsymbol{\nu} a)(P \mid a^m \multimap \bar{b}^n) \vdash \Gamma, \bar{b} : A^\perp}\,(\text{cut}) \qquad \rightarrow^c_{sc} \begin{array}{c} \Pi[\bar{b}^n/\bar{a}^\ell] \\ \vdots \\ P[\bar{b}^n/\bar{a}^\ell] \vdash \Gamma, \bar{b} : A^\perp \end{array}$$

commutation steps. If one of the premisses is not an introduction rule for the cut formula, then the cut can be permuted with the last rule of this premiss.

commutativity steps. The premisses of a cut rule may always be permuted.

In the first two cases, two name occurrences are eliminated, then the annotation c is the involution that associates them. In the last two cases, c is empty.

Definition 11. *The* annotated cut elimination *relation $\rightarrow^c_{sc}$ over type derivations is the reflexive transitive closure the rules above in all contexts. In the transitive closure, for $\Pi \rightarrow^c_{sc} \Pi' \rightarrow^d_{sc} \Pi''$ we define $\Pi \rightarrow^{c\cup d}_{sc} \Pi''$.*

We recall that $\mathrm{MLL_u}$ sequent calculus proofs, as well as type derivations, admit a standard cut-elimination theorem using the rules of this definition.

Theorem 1 (subject reduction). *For all type derivation $\Pi : P \vdash \Gamma$ and execution step $P \rightarrow^c_{ex} P'$, there is a derivation $\Pi' : P' \vdash \Gamma$ such that $\Pi \rightarrow^c_{sc} \Pi'$.*

Proof (sketch). The first step is to remark that, for a given execution step $P \rightarrow^c_{ex} P'$, we can use commutation of cuts and commutativity (which relate structurally congruent terms) to get the type derivation of P in a canonical form

$$P \equiv C[(\boldsymbol{\nu} a_n)(\cdots(\boldsymbol{\nu} a_1)((\boldsymbol{\nu} a_0)(\bar{a}_0(\zeta).S \mid a_0 \multimap \bar{a}_1) \mid a_1 \multimap \bar{a}_2) \cdots \mid a_n(\bar{\zeta}).T)]$$

where C is a context made of cut rules. Then we use the axiom rules to eliminate the forwarders and the multiplicative rule to eliminate the actions.

Corollary 2. *All typable terms are alive.*

Proof (sketch). Since we consider terms without replication, all execution sequences are finite, so by subject reduction we can consider terms irreducible by execution. Then we can proceed by induction on the type derivation.

Definition 12. *Let $\Gamma = \{x_i : A_i\}_{i=1}^n$ be a type. A context for type Γ is a family of typed terms $T_i \vdash \Delta_i, \bar{x}_i : A_i^\perp$ that exhibit actions on the $\bar{x}_i$. For any term P, define $C[P] = (\boldsymbol{\nu} x_1 \dots x_n)(P \mid T_1 \mid \dots \mid T_n)$.*

Definition 13 (extended execution). *Let $\sim$ be the equivalence generated by the following rules in all contexts, assuming $(\xi \cup \{\bar{y}\}) \cap FV(Q) = \emptyset$:*

$$x(\zeta).(y(\xi).P \mid Q) \sim y(\xi).x(\zeta).(P \mid Q) \qquad (\boldsymbol{\nu} a)(y(\xi).P \mid Q) \sim y(\xi).(\boldsymbol{\nu} a)(P \mid Q)$$

Extended reduction $\rightarrow_{xx}$ is the relation such that $P \rightarrow_{xx}^c Q$ iff $P \sim \rightarrow_{ex}^c \sim Q$.

Proposition 7. *For all terms $P \sim Q$, and type Γ, $P \vdash \Gamma$ iff $Q \vdash \Gamma$. Then, for all context $C[]$ for type Γ, $C[P]$ and $C[Q]$ have the same consistent pairings.*

Proof (sketch). For type derivations, these rules are commutations of $\parr$ introductions with $\otimes$ introductions and cuts, which are correct in sequent calculus. Pairings are obviously preserved, and consistent ones are preserved because of the structure of cuts: exchanging two prefixes that are both cut does not change pairings since the processes they are cut against are necessarily independent.

5 Wireless proof nets

Definition 14 (paired graph). *A* paired graph *$G = (V, E \uplus A, P)$ is a mixed simple graph $(V, E \uplus A)$ with undirected edges E, directed edges (or* arcs*) A, plus a set of paired arcs $P \subset \mathscr{P}(A)$ which are pairwise disjoint sets of arcs with the same end node, called a* pairing node*. An* ordered *paired graph is a paired graph where each node v is equipped a total order $\leq_v$ over its incoming edges.*

Paths, cycles and the complement of a paired graph are standard definitions.

Definition 15 (wireless proof net). *A wireless proof structure (WPS) R consists of an ordered paired graph $G_R = (V, E \uplus A, P)$ where each node has a sort in $\{ax, \otimes, \parr\}$ and an MLL_u formula as label, with two disjoint sets of nodes C^R (the conclusions) and $\mathscr{C}^R$ (the cut nodes). A wireless proof net (WPN) is a structure built by the following MLL_u sequent calculus rules:*

Ax: *$(\{u, v\}, \{uv\} \uplus \emptyset, \emptyset)$ is a WPN, called axiom-link, with two nodes of sort ax labelled respectively A and $A^\perp$. Both nodes are conclusions but not cut-nodes.*

$\parr$: *If $G = (V, E \uplus A, P)$ is a WPN with conclusions $C \cup \{x_i : A_i\}_{i=1}^n$ and cut-nodes $\mathscr{C}$, then a WPN is obtained by extending G with a pairing node y of sort $\parr$, labelled $\parr_{i=1}^n A_i$, adjacent with new paired arcs from $(x_i)_{i=1}^n$ in this order, with $C \cup \{y\}$ as conclusions and the same cut node set.*

$\otimes$: *If $\{G_i = (V_i, E_i \uplus A_i, P_i)\}_{i=1}^n$ are disjoint WPNs with respective conclusions $C^i \cup \{x_i : A_i\}$ and cut-nodes $\mathscr{C}^i$, then $\biguplus_{i=1}^n G_i$ extended with a node y of sort $\otimes$, labelled $\bigotimes_{i=1}^n A_i$, adjacent with new arcs from $(x_i)_{i=1}^n$ in this order, is a WPN with conclusions $\bigcup_{i=1}^n C^i \cup \{y\}$ and cut-nodes $\bigcup_{i=1}^n \mathscr{C}^i$.*

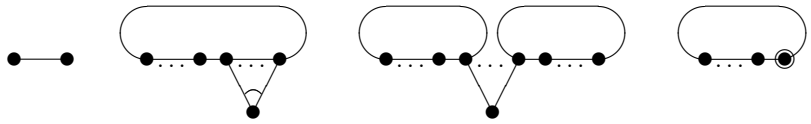


Fig. 1. Representation of WPNs: axiom link, $\parr$ node, $\otimes$ node and cut nodes.

Cut: *If* $\{G_i = (V_i, E_i \uplus A_i, P_i)\}_{i\in\{1,2\}}$ *are two disjoint WPNs with respective conclusions* $C^i \cup \{x_i : A_i\}$ *and cut-nodes* $\mathscr{C}^i$ *and such that* $A_1 = A_2^\perp$*, then* $G_1 \uplus G_2$ *is a WPN with conclusions* $C^1 \cup C^2$ *and cut-nodes* $\mathscr{C}^1 \cup \mathscr{C}^2 \cup \{x_1, x_2\}$*.*

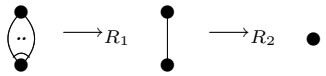
A standard proof net is defined as a WPN except that there is no cut-node set, instead in the cut rule the graph is extended with an edge $\{x_1, x_2\}$, distinguished from axiom-links, usually called a *wire*.

When a WPN (respectively a proof net) is induced from a type derivation Π, we write $[\![\Pi]\!]$ (respectively $[\Pi]$). By definition the conclusion set of $[\![\Pi]\!]$ is labelled by Γ, the cut-node set of $[\![\Pi]\!]$ is labelled with the cut formulas of the type derivation Π and pairing nodes are the set of $\parr$ rules in Π.

Remark that given a typed term $P \vdash \Gamma$, if $\{\Pi_i : P' \vdash \Gamma\}_{i\in I}$ is the set of type derivations of terms $P' \equiv P$ with conclusion Γ then all the Π_i induce the same WPN. It is also the same WPN up to type substitution for all typings of P.

The graphical notation of WPNs is presented in figure 1. By definition there are undirected edges only in axiom links, moreover WPNs can be drawn considering the up-bottom orientation of arcs, so we keep arc orientation implicit by this convention. Paired arcs are joint by a circle arc on the side of the pairing node. By definition only conclusion nodes and cut nodes have outdegree 0, we distinguish cut nodes by drawing circles around them. Because of the building rules, node sorts can be deduced from labels, except for axiom links, which are identified by adjacent undirected edges, moreover the conclusion labels suffice to deduce all labels, so we keep most of this information implicit.

Definition 16 (contractile paired graph). *A paired graph is* contractile *if it rewrites to a single node without edges or arcs by the following two rules, where nodes are distinct and* R_2 *applies only to non paired edges:*



This definition is one of the standard correctness criteria for proof nets [2].

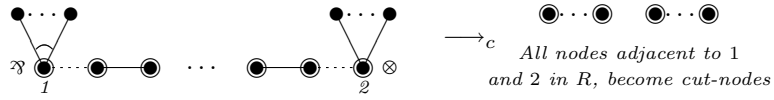
The set of (unordered) pairs of nodes of a wireless proof net R that may correspond to $\mathrm{MLL_u}$ binary sequent calculus cuts, contains no edges of R but edges in the complement graph of R. This is the following set of cut-edges:

Definition 17 (cut-edge). *Let* A *be a* $\mathrm{MLL_u}$ *formula labelling some node in a WPN* R*. Call* L_A *the set of edges of axiom-links labelled* A *or* $A^\perp$ *in* R*. The set* $\mathscr{E}_A^R$ *of* cut-edges *of label* A *is the set of pairs* $\{u, v\}$*, with* $u, v \in \mathscr{C}^R$ *respectively labelled* $A, A^\perp$ *such that* $uv \notin L_A$*.*

Definition 18 (cut-path). *Let A be a* $\mathrm{MLL_u}$ *formula labelling some node in a WPN R. A* cut-path *of label A is a non-empty acyclic path whose internal nodes are of sort ax and terminal nodes are of dual sorts $\otimes$ and $\parr$, such that 1. non ending edges alternate in L_A and in $\mathscr{E}_A^R$, 2. ending edges are in $\mathscr{E}_A^R$.*

Cut-edges are drawn with dashed lines to distinguish them from edges in R. A path reduction in a WPN corresponds, in sequent calculus, to a sequence a logical cut elimination steps for axioms followed by a multiplicative step:

Definition 19 (path reduction). *Let ρ be a cut-path of label A in R. A* path reduction $\rightarrow_c$ *of ρ is a reduction of redex ρ whose reduct is obtained by erasing all nodes of ρ as follows:*



Definition 20 (non-deterministic reduction). *Let R be a WPN. Let $\mathscr{F}_R$ be the set of formulas A labelling R for which $\mathscr{E}_A^R \neq \emptyset$. We call* cut-choice *in R a set C of cut-paths of distinct labels $A \in \mathscr{F}_R$ such that the paired graph $R \cup C$ is contractile. We write $\mathbb{C}_R$ the set of cut-choices in R.* WPN reduction *$R \Rightarrow_{C_A} R'$ is defined by a cut-choice C_A of R containing a cut-path ρ of label A whose set of nodes is the redex. The reduct is obtained by path reduction of ρ.*

Definition 21 (permutation equivalence). *Two WPNs R, S are equivalent, written $R \simeq S$, if they are equal up to the ordering of edges and for all x and i, the source of the i-th incoming edge of x in R and in S have the same label.*

Proposition 8 (bisimilarity). *If $R \simeq S$ and $R \Rightarrow_{C_A} R'$ then there is an S' such that $S \Rightarrow_{C_A} S'$ and $R' \simeq S'$.*

6 Translation of processes

Definition 22 (translation). *The translation $[\![P :: \mathcal{I}]\!]$ of an interfaced term is a WPS in which the node set is $\mathcal{L}(P)$, sorts and edges are defined as follows:*

- *For all forwarder $a^\ell \multimap \bar{b}^m$ occurring in P, put an undirected edge between ℓ and m, both of which are of sort ax.*
- *For all action $x^\ell(z_1 \dots z_k).Q$ occurring in P, put an arc (m, ℓ) for each m such that* $\mathrm{subj}\, m = z_i$ *for some i, and order these arcs in any order such that for all $i < j$, $(m, \ell) < (n, \ell)$ if* $\mathrm{subj}\, m = z_i$ *and* $\mathrm{subj}\, n = z_j$*. If* $\mathrm{pol}\, x = \uparrow$*, then ℓ has sort $\parr$ and these arcs form a pair and ℓ is a pairing node, otherwise the sort of ℓ is $\otimes$.*

The label of a node ℓ is $[\![\iota]\!]$ if ι is the sort of $\mathrm{subj}\, \ell$*, the conclusion set is the set of locations of free names in P, and the cut node set is the set of locations of hidden names in P. Figure 2 illustrates this translation.*

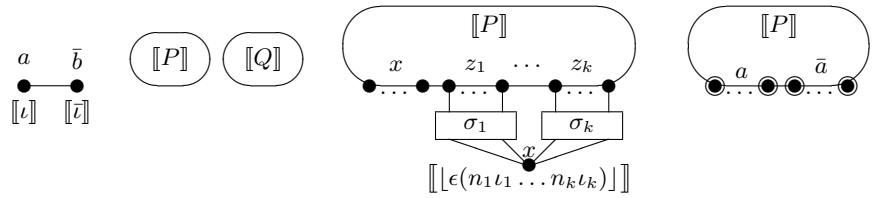


Fig. 2. Translated forwarder, parallel composition, action and hiding. The σ_i are arbitrary permutations of n_i

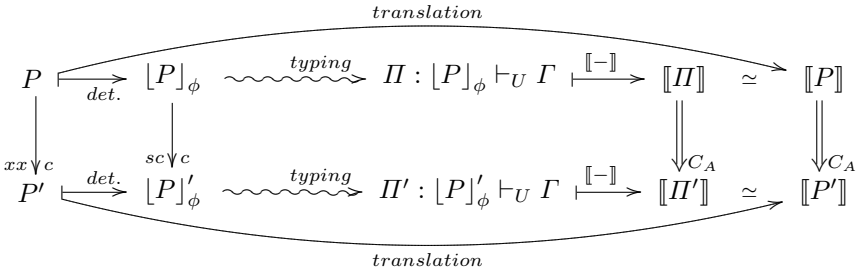
Note in particular that the translation is only defined up to some class of permutations, which corresponds to permutation equivalence. The definition above can also be formulated in an inductive way, from which we easily deduce that for a type derivation $\Pi : P \vdash [\![\mathcal{I}]\!]$, $[\![P :: \mathcal{I}]\!]$ is a WPN of conclusion $[\![\mathcal{I}]\!]$.

Definition 23. *An interfaced term $P :: \mathcal{I}$ is called* correct *if $[\![P]\!]$ is a WPN.*

We now formulate our main result, which is the precise correspondence in dynamics between correct terms and wireless proof nets. Correspondence is up to commutation of actions, which is an operation that is not observable by typed contexts, as stated in proposition 7.

Theorem 2. *Let $P :: \mathcal{I}$ be an interfaced term. Let U by the equivalence relation $\{(\ell, m); \mathrm{subj}\,\ell = \mathrm{subj}\,m$ or $\mathrm{subj}\,\ell = \overline{\mathrm{subj}\,m}\}$. For all numbering ϕ of P and type derivation $\Pi : \lfloor P \rfloor_\phi \vdash_U \Gamma$, $[\![\Pi]\!] \simeq [\![P]\!]$. If P is correct, then for all reduction $[\![P]\!] \Rightarrow_{C_A} R$ there is an extended execution step $P \rightarrow_{xx} P'$ such that $[\![P']\!] = R$.*

The first point is a simple verification. For the second one, the existence of a WPN reduction implies the existence of a cut choice, which induces a proof net, which can be sequentialised into a typing derivation Π for a determinisation $\lfloor P \rfloor_\phi$, using commutation as of definition 13. Extended execution in turn implements all proof net reductions. The following diagram sums up the argument:



Hence cut choices correspond to deadlock-free executions, and WPN reduction steps implement the steps of these executions. The deadlock-freeness problem of concurrent processes is thus translated into a correctness problem in wireless proof structures of $\mathrm{MLL_u}$. We hope to find an efficient way to do this as deciding the correctness in standard proof structures is NL-complete [7].

References

1. Emmanuel Beffara. A concurrent model for linear logic. In *21st International Conference on Mathematical Foundations of Programming Semantics (MFPS)*, volume 155, pages 147–168, may 2006.
2. Vincent Danos. *La logique linéaire appliquée à l'étude de divers processus de normalisation (et principalement du λ-calcul)*. PhD thesis, Univ. Paris VII, 1990.
3. Thomas Ehrhard and Olivier Laurent. Interpreting a finitary π-calculus in differential interaction nets. In Luís Caires and Vasco T. Vasconcelos, editors, *18th International Conference on Concurrency Theory (Concur)*, volume 4703 of *LNCS*, pages 333–348. Springer, September 2007.
4. Thomas Ehrhard and Laurent Regnier. Differential interaction nets. In *Workshop on Logic, Language, Information and Computation*, 2004. Invited paper.
5. Jean-Yves Girard. Proof-nets : the parallel syntax for proof theory. *Logic and Algebra*, 180, 1996.
6. Kohei Honda and Olivier Laurent. An exact correspondence between a typed π-calculus and polarised proof-nets. Submitted, November 2008.
7. Paulin Jacobé de Naurois and Virgile Mogbil. Correctness of multiplicative additive proof structures is nl-complete. In *Proceedings of the twenty-third annual IEEE symposium on logic in computer science (LICS)*, pages 476–485, 2008.
8. Harry G. Mairson and Kazushige Terui. On the computational complexity of cut-elimination in linear logic. In Carlo Blundo and Cosimo Laneve, editors, *Theoretical Computer Science, 8th Italian Conference, ICTCS*, volume 2841 of *Lecture Notes in Computer Science*, pages 23–36. Springer, 2003.
9. François Maurel. Nondeterministic light logics and NP time. In Martin Hofmann, editor, *Typed Lambda Calculi and Applications (TLCA), 6th International Conference*, number 2701 in LNCS, pages 241–255. Springer, 2003.
10. Virgile Mogbil. Parallel and non-deterministic linear logic. Unpublished manuscript, Sept. 2009.
11. Kazushige Terui. Proof nets and boolean circuits. In *19th IEEE Symposium on Logic in Computer Science (LICS)*, pages 182–191, 2004.
12. Nobuko Yoshida, Martin Berger, and Kohei Honda. Strong normalisation in the π-calculus. In *16th IEEE Symposium on Logic in Computer Science (LICS)*, pages 311–322, 2001.

A Proofs on the calculus and determinization

Lemma 1

Proof. First remark that, by definition of execution, we have $\mathcal{L}(P) = \mathcal{L}(Q) \uplus \operatorname{dom} c$, besides $\operatorname{dom} d \subset \mathcal{L}(Q)$ so the domains of c and d are disjoint. We can thus define the involution $c' = c \cup d$, and we check that it is indeed a pairing of P.

Let $\ell \in \operatorname{dom} c'$. If $\ell \in \operatorname{dom} c$ then ℓ is the location of an action involved in the execution step, so $\operatorname{subj}_P \ell$ is a hidden name and we have $\operatorname{subj}_P c(\ell) = \overline{\operatorname{subj}_P \ell}$ by definition of execution, and subsequently we get $\operatorname{subj}_P c'(\ell) = \overline{\operatorname{subj}_P \ell}$.

Otherwise ℓ is in the domain of d. If $\operatorname{subj}_Q \ell$ is action-bound in Q, then there is a location $m \in \mathcal{L}(Q)$ and an integer i such that $\operatorname{subj}_Q \ell = \operatorname{obj}^i_Q m$ and $\operatorname{subj}_Q d(\ell) = \operatorname{obj}^i_Q d(m)$. In this case, we can easily check that the property also holds in P, that is $\operatorname{subj}_P \ell = \operatorname{obj}^i_P m$ and $\operatorname{subj}_P c'(\ell) = \operatorname{obj}^i_P c'(m)$. If, on the contrary, $\operatorname{subj}_Q \ell$ is hidden in Q, then two sub-cases may occur for $\operatorname{subj}_P \ell$: either it is also hidden or it is bound by an action labelled by some some location m in the domain of c. In the first sub-case, by definition of execution, the same property holds for $d(\ell)$, so we have $\operatorname{subj}_P c'(\ell) = \overline{\operatorname{subj}_P \ell}$. In the second sub-case, the only way we can have $\operatorname{subj}_Q d(\ell) = \overline{\operatorname{subj}_Q \ell}$ is if there is an integer i such that $\operatorname{subj}_P \ell = \operatorname{obj}^i_P m$ and $\operatorname{subj}_P d(\ell) = \operatorname{obj}^i_P c(m)$, hence $\operatorname{subj}_P c'(\ell) = \operatorname{obj}^i_P c'(m)$.

Hence c' is a correct pairing pairing of P.

Now suppose d that is consistent but c' is not. Then there exists a cycle $\ell_0 \prec_{c'} \ell_1 \prec_{c'} \cdots \prec_{c'} \ell_k = \ell_0$. If all the ℓ_i are in Q then this is a cycle in $\prec_d$, which cannot be. Since c annotates a reduction of P, all elements of $\operatorname{dom} c$ are minimal for $\leq_P$, so the cycle cannot consist only of elements of $\operatorname{dom} c$. So we may assume $\ell_0 \in Q$ and $\ell_1 \in \operatorname{dom} c$. This means either $\ell_0 <_P \ell_1$ or $c'(\ell_0) < c'(\ell_1)$, in each case this implies that some location in $\operatorname{dom} c$ is prefixed, which is impossible. Hence c' is consistent.

Lemma 2

Proof. By the existence of the execution step $P \rightarrow^{c_1}_{ex} Q_1$, we know that P can be written

$$P \equiv (\boldsymbol{\nu u})\Big(\bar{a}^{\ell_0}_0(\bar{\zeta}).S \mid (a^{m_{i-1}}_{i-1} \multimap \bar{a}^{\ell_i}_i)^n_{i=1} \mid a^{m_n}_n(\zeta).T \mid P'\Big)$$

such that $\operatorname{dom} c_1 = \{\ell_i, m_i\}^n_{i=0}$. The term P can be decomposed in a similar way to jusitfy the execution step $P \rightarrow^{c_2}_{ex} Q_2$:

$$P \equiv (\boldsymbol{\nu u})\Big(\bar{b}^{\ell'_0}_0(\bar{\zeta}').S' \mid (b^{m'_{i-1}}_{i-1} \multimap \bar{b}^{\ell'_i}_i)^{n'}_{i=1} \mid b^{m'_{n'}}_{n'}(\zeta').T' \mid P''\Big)$$

with $\operatorname{dom} c_2 = \{\ell'_i, m'_i\}^{n'}_{i=0}$. By hypothesis the domains of c_1 and c_2 are disjoint, so all the ℓ_i and m_i are distinct from all the ℓ'_j and m'_j. As a consequence the

term P can be decomposed as

$$P \equiv (\boldsymbol{\nu u})\Big(\bar{a}_0^{\ell_0}(\bar{\zeta}).S \mid (a_{i-1}^{m_{i-1}} \multimap \bar{a}_i^{\ell_i})_{i=1}^{n} \mid a_n^{m_n}(\zeta).T \\ \mid \bar{b}_0^{\ell'_0}(\bar{\zeta}').S' \mid (b_{i-1}^{m'_{i-1}} \multimap \bar{b}_i^{\ell'_i})_{i=1}^{n'} \mid b_{n'}^{m'_{n'}}(\zeta').T' \mid P'''\Big)$$

and the terms Q_1 and Q_2 have execution steps with the expected annotations, with the common reduct $R = (\boldsymbol{\nu u}\zeta\zeta')(S \mid T \mid S' \mid T' \mid P''')$. Unicity of R up to structural congruence is a consequence of the fact the c_1 and c_2 completely describe which subterms of P, Q_1, Q_2 interact and in which way.

Proposition 9. *Let P_0 be a term. For any two execution sequences*

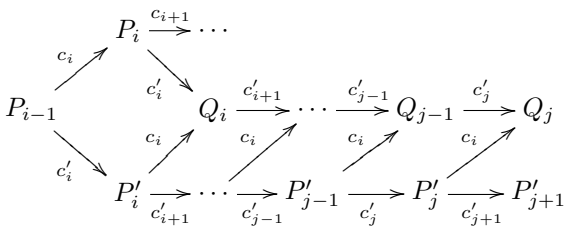
$$P_0 \to_{ex}^{c_1} P_1 \to_{ex}^{c_2} \cdots \to_{ex}^{c_m} P_m P_0 \to_{ex}^{c'_1} P'_1 \to_{ex}^{c'_2} \cdots \to_{ex}^{c'_n} P'_n \qquad \textit{with} \qquad \bigcup_{i=1}^{m} c_i = \bigcup_{j=1}^{n} c'_j$$

we have $P_m \equiv P'_n$. Moreover, $m = n$ and there is a permutation σ of $\{1, \ldots, n\}$ such that for all i, $c'_i = c_{\sigma(i)}$.

Proof. We first establish that $m = n$ and that the sequences are permutations of each other. Let $c = \bigcup_{i=1}^{m} c_i = \bigcup_{j=1}^{n} c'_j$ be the pairing induced by the considered reduction sequences. Consider an index i. By construction, the set $\operatorname{dom} c_i$ can be written $\{\ell_k; 1 \leq k \leq 2p\}$ for some integer $p \geq 1$, so that ℓ_1 is the location of a negative action, ℓ_p is that of a positive action, for each $k < p$ there is forwarder from ℓ_{2k} to ℓ_{2k+1}, and for each $k \leq p$ we have $c(\ell_{2k-1}) = \ell_{2k}$. Since $\ell_1 \in \operatorname{dom} c_i \subseteq \operatorname{dom} c$, there is a j such that $\ell_1 \in \operatorname{dom} c'_j$. Then we must have $c'_j(\ell_1) = \ell_2$. For each $k < p$ such that $\ell_{2k} \in \operatorname{dom} c'_j$, since the location ℓ_{2k} is that of one side of a forwarder, necessarily $\operatorname{dom} c'_j$ contains the location of the other side, namely ℓ_{2k+1}. Then $\ell_{2p-1} \in \operatorname{dom} c'_j$, and $c'_j(\ell_{2p-1}) = c(\ell_{2p-1}) = c_i(\ell_{2p-1}) = \ell_{2p}$. As a consequence, we have $c_i = c'_j$. By the same argument, for each j there is an i such that $c'_j = c_i$, besides the sequences (c_i) and (c'_j) consist of pairwise distinct pairings, so this estabishes a permutation σ between the two sequences, and subsequently $m = n$.

We now prove that the final terms P_n and P'_n are equal (up to structural congruence). Call $d(\sigma)$ the number of pairs (i, j) such that $i < j$ and $\sigma(i) > \sigma(j)$. We proceed by induction on $d(\sigma)$. If this number is 0, then σ is the identity function and the sequences match, so obviously we have $P_n = P'_n$. Otherwise, consider a minimal i such that $\sigma(i) \neq i$, hence $\sigma^{-1}(i) \neq i$ and $\sigma^{-1}(i) > i$, and let $j = \sigma^{-1}(i) - 1$. The reduction sequences match in their first $i-1$ steps, then one has a reduction labelled c_i while the other has a reduction labelled $c'_i = c_{\sigma(i)}$. By repeated applications of lemma 2, we can deduce that for each $k \geq i$ there is

a term Q_k such that $P'_k \to_{ex}^{c_i} Q_k$ and $Q_{k-1} \to_{ex}^{c'_k} Q_k$ if $k > i$:



Moreover, by construction $c'_{j+1} = c_{\sigma(j+1)} = c_i$, so $P'_{j+1} = Q_j$, because there is at most one possible reduction for a given annotation. Hence we can deduce a pair of reduction steps $P'_{j-1} \to_{ex}^{c'_{j+1}} Q_{j-1} \to_{ex}^{c'j} P_{j+1}$. This yields a new reduction sequence from P_0 to P'_n that corresponds to a new permutation σ' of the sequence (c_i), and σ' is σ where $\sigma(j)$ and $\sigma(j+1)$ are swapped. By definition of j we have $\sigma(j) > \sigma(j+1)$ so $\sigma'(j) < \sigma'(j+1)$. For any $a \notin \{j, j+1\}$ we have $\sigma(a) < \sigma(j)$ if and only if $\sigma'(a) < \sigma'(j+1)$, and the same exchanging j and $j+1$, so we have $d(\sigma') = d(\sigma) - 1$, and we can conclude by induction hypothesis.

Proposition 10 (canonical form). *Any term P can be decomposed as*

$$P \equiv (\boldsymbol{\nu} a_1) \cdots (\boldsymbol{\nu} a_n)\Big(\prod_{i=1}^{k} x_i(\zeta_i).P_i \;\Big|\; \prod_{j=1}^{p} b_j \multimap \bar{c}_j\Big)$$

where the product stands for any parallel composition of a family of terms.

Proof. We proceed by induction on the structure of the term P. For parallel composition, we use the scope extension rule to get all the hidings at top level, then associativity and commutativity of parallel compositions allow the necessary reordering of actions and forwarders. All other cases are immediate.

Proposition 11. *Interfaces and determinism are preserved by structural congruence and execution.*

Proof. Preservation of interfaces by the commutative monoid laws for parallel composition are immediate, since parallel composition reduces to summation of muliplicities in interfaces. Commutation of hidings on different names is immediate. For the scope extrusion and garbage collection rules, remark that in an interfaced term $P :: \mathcal{I}$, the free names of P are exactly those appearing with a non-null multiplicity in $\mathcal{I}$. Therefore if a is a name such that neither a nor $\bar{a}$ are free in P, then we may assume that $\mathcal{I}$ does not contain a sort assignement for a and $\bar{a}$, then we can write $P :: \mathcal{I}, a : 0{\downarrow}I, \bar{a} : 0{\uparrow}\bar{I}$, on which we can apply the hiding rule to get $(\boldsymbol{\nu} a)P :: \mathcal{I}$.

Let us now consider an instance of the execution rule $T \to_{ex} T'$, with the notations of definition 4. Suppose that the left-hand side is interfaced. Then, in

the derivation of this interface, there are interfaces $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$ and a channel sort I such that the sub-terms are interfaced as

$$P :: \mathcal{I}, \bar{a}_0 : m{\uparrow}\bar{I}, \bar{\zeta} : \bar{I} \qquad Q :: \mathcal{J}, a_n : p{\downarrow}I, \zeta : I \qquad R :: \mathcal{K}$$

and subsequently $\bar{a}_0(\bar{\zeta}).P \mid (a_{i-1} \multimap \bar{a}_i)_{i=1}^n \mid a_n(\zeta).Q \mid R$ has interface

$$\mathcal{I} + \mathcal{J} + \mathcal{K} + \big(\bar{a}_0 : (m+1){\uparrow}\bar{I}, \{a_{i-1} : 1{\downarrow}I, \bar{a}_i : 1{\uparrow}\bar{I}\}_{i=1}^n, a_n : (p+1){\downarrow}I\big)$$

where the sum notation represents the operation of adding multilicities for compatible interfaces, as in the interface rule for parallel composition.

If we do not assume that T is deterministic, then the interfaces $\mathcal{I}$, $\mathcal{J}$ and $\mathcal{K}$ may contain assignements for the names a_i and $\bar{a}_i$, but the presence of the $(\boldsymbol{\nu} a_i)$ in the complete term impose that the number of occurrences of a_i and $\bar{a}_i$ match for each i (with independent multiplicities for different values of i). In the reduct, we deduce the interface as

$$P \mid Q \mid R :: \mathcal{I} + \mathcal{J} + \mathcal{K} + \big(\bar{a}_0 : m{\uparrow}\bar{I}, a_n : p{\downarrow}I, \zeta : I, \bar{\zeta} : \bar{I}\big)$$

in which, for each i, the multiplicities of a_i and $\bar{a}_i$ match (indeed they are one less as in the previous term). The multiplicities and sorts also match for the fresh names in the sequence ζ, therefore the hidings on $(\boldsymbol{\nu}\zeta)$ and $(\boldsymbol{\nu} a_i)$ are admissible. The interface of $(\boldsymbol{\nu} a_0 \dots a_n)\big((\boldsymbol{\nu}\zeta)(P \mid Q) \mid R\big)$ is the same as that of the initial term, namely the sum $\mathcal{I}+\mathcal{J}+\mathcal{K}$ with any assignement to the a_i and $\bar{a}_i$ removed.

If T is assumed to be deterministic, then in the previous argument we must have $m = p = 0$, and the interfaces $\mathcal{I}$, $\mathcal{J}$ and $\mathcal{K}$ may only contain assignements for the a_i and $\bar{a}_i$ with multiplicitiy 0. In T', we thus have multiplicity 0 for all the a_i and $\bar{a}_i$, so T' is determinstic too.

Proposition 2

Proof. Let P be a determinstic term. The set of pairings is not empty, since the empty function is always a pairing. Suppose we can find two distinct maximal pairings c and d. The set of locations on which c and d do not coincide is not empty (and obviously finite), so we can consider an element ℓ of this set that is minimal for the prefixing order $\leq_P$. At least $c(\ell)$ or $d(\ell)$ is defined, we assume it is c without loss of generality. Two cases may occur, depending on whether $\operatorname{subj} c(\ell)$ is action-bound or hidden.

First suppose it is action-bound, then there is a location m and an integer i such that $\operatorname{subj} \ell = \operatorname{obj}^i m$ and $\operatorname{subj} c(\ell) = \operatorname{obj}^i c(m)$. If $d(\ell)$ is also defined, we must also have $\operatorname{subj} d(\ell) = \operatorname{obj}^i d(m)$, and by the minimality of ℓ we must have $c(m) = d(m)$, hence $\operatorname{subj} d(\ell) = \operatorname{subj} c(\ell)$. By the determinism hypothesis, the name $\operatorname{subj} c(\ell)$ occurs exactly once in P, so we must have $d(\ell) = c(\ell)$, which is contradictory with the definition of ℓ. Therefore $d(\ell)$ cannot be defined.

Similarly, if $d(c(\ell))$ is defined, since $\operatorname{subj} c(\ell) = \operatorname{obj}^i c(m)$, then we must have $\operatorname{subj} d(c(\ell)) = \operatorname{obj}^i d(c(m))$, but since $c(m) = d(m)$ we have $d(c(m)) = d(d(m)) = m$, hence $\operatorname{subj} d(c(\ell)) = \operatorname{obj}^i m = \operatorname{subj} \ell$. By the determinism hypothesis this implies that $d(c(\ell)) = \ell$, an by involutivity of d this implies that

$c(\ell) = d(\ell)$, which is contradictory. Hence both $d(\ell)$ and $d(c(\ell))$ are undefined. We can thus extend d by setting $d(\ell) = c(\ell)$, which yields a larger pairing than d, contradicting the maximality of d.

As a consequence, $\mathrm{subj}\, c(\ell)$ cannot be action-bound, so we have $\mathrm{subj}\, c(\ell) = \overline{\mathrm{subj}\, \ell}$. By the determinism hypothesis, the names $\mathrm{subj}\, \ell$ and $\mathrm{subj}\, c(\ell)$ have exactly one occurrence each, at respective locations ℓ and $c(\ell)$. Therefore, either $d(\ell)$ is defined and equal to $c(\ell)$, or $d(\ell)$ and $d(c(\ell))$ are undefined. The former contradicts the minimality of ℓ, the latter contradicts the maximality of d.

Hence $c = d$ and we get the uniqueness of the maximal pairing.

Proposition 12. *Execution in deterministic terms is strongly confluent.*

Proof. Let P be a deterministic term, in a canonical form using the notations of proposition 10. Consider a pair of distinct execution steps $P \to_{ex} Q$ and $P \to_{ex} Q'$. We prove that the set of actions and forwarders involved in these executions are disjoint. Let $m \in \mathbb{N}$ and $f : [1, m] \to [1, n]$ be such that the execution $P \to_{ex} Q$ involves a negative action on $\bar{a}_{f(1)}$, forwarders $a_{f(i)} \multimap \bar{a}_{f(i+1)}$ for each $1 \leq i < m$ and a positive action on $a_{f(m)}$. Define m' and f' for the execution $P \to_{ex} Q'$ similarly. For contradiction, suppose there are j and j' such that $f(j) = f'(j')$. By hypothesis the term P contains exactly one occurrence of the name $a_{f(j)}$. If this occurrence is a forwarder $a_{f(j)} \multimap \bar{a}_k$ then we have $m > j$, $m' > j'$ and $f(j+1) = f'(j'+1) = k$, otherwise it is an action and we have $m = j$ and $m' = j'$. Similarly, P contains exactly one occurrence of $\bar{a}_{f(j)}$. If this occurrence is a forwarder $a_k \multimap \bar{a}_{f(j)}$ then we have $j > 1$, $j' > 1$ and $f(j-1) = f'(j'-1) = k$, otherwise it is an action and we have $j = j' = 1$. From this we deduce that if the codomains of f and f' intersect, then $m = m'$ and $f = f'$, and the executions $P \to_{ex} Q$ and $P' \to_{ex} Q'$ are actually the same. Hence these reductions involve disjoints parts of the term P, so we can find a decomposition $P = (\boldsymbol{\nu} a_1 \dots a_n)(P_1 | P_2)$ such that $P \to_{ex} Q$ is a reduction $P \to_{ex} (\boldsymbol{\nu} a_1 \dots a_n)(P_1' \mid P_2)$ and $P \to_{ex} Q'$ is a reduction $P \to_{ex} (\boldsymbol{\nu} a_1 \dots a_n)(P_1 \mid P_2')$. By setting $R = (\boldsymbol{\nu} a_1 \dots a_n)(P_1' \mid P_2')$ we get the execution steps $Q \to_{ex} R$ and $Q' \to_{ex} R$, which proves strong confluence.

Proposition 3

Proof. For a subterm Q of P, a name x and an integer i, let $\delta(Q, x, i)$ be the number of locations $\ell \in \mathcal{L}(Q)$ such that $\mathrm{subj}\, \ell = x$ and $\phi(\ell) = i$ (hence 0 or 1). We actually prove, by induction on the interface derivation of P, that for all subterm Q of interface $\mathcal{I} := \{x_i : n_i \iota_i\}_{1 \leq i \leq k}$ the term $\lfloor Q \rfloor_\phi$ has interface $\lfloor \mathcal{I} \rfloor_Q := \left\{ x_i^j : \delta(Q, x_i, j) \lfloor \iota_i \rfloor \right\}_{\substack{1 \leq i \leq k \\ 1 \leq j \leq \sharp(P, x_i)}}$.

Consider a subterm $Q' = x^\ell(z_1 \dots z_n).Q$ of P. By induction we have an interface derivation that ends as

$$\frac{Q :: \mathcal{I}, x : n\epsilon(n_1 \iota_1 \dots n_k \iota_k), \{z_i : n_i \iota_i\}_{i=1}^k \quad \epsilon = \mathrm{pol}\, x}{x^\ell(z_1 \dots z_n).Q :: \mathcal{I}, x : (n+1)\epsilon(n_1 \iota_1 \dots n_k \iota_k)}$$

By induction hypothesis the term $\lfloor Q \rfloor_\phi$ has interface

$$\lfloor \mathcal{I} \rfloor_Q , \left\{ x^j : \delta(Q,x,j)\epsilon\big((1\lfloor \iota_1 \rfloor)^{n_1} \dots (1\lfloor \iota_k \rfloor)^{n_k}\big) \right\}_{1\le j\le \sharp(P,x)}, \left\{ z_i^j : \delta(Q,z_i,j)\lfloor \iota_i \rfloor \right\}_{\substack{1\le i\le k \\ 1\le j\le \sharp(P,z_i)}}$$

The names z_i are bound by the action $x^\ell(z_1 \dots z_n)$, so for each i, all n_i occurrences of z_i in P are in Q, therefore for all j we have $\delta(Q,z_i,j) = 1$. Besides, by the injectivity of $\ell \mapsto (\operatorname{subj} \ell, \phi(\ell))$, we have $\delta(Q,x,\phi(\ell)) = 0$ and $\delta(Q',x,\phi(\ell)) = 1$, and $\delta(Q',y,j) = \delta(Q,y,j)$ when $(y,j) \neq (x,\phi(\ell))$ and y is not one of the z_i. Therefore Q' has interface

$$\lfloor \mathcal{I} \rfloor_{Q'} , \left\{ x^j : \delta(Q',x,j)\epsilon\big((1\lfloor \iota_1 \rfloor)^{n_1} \dots (1\lfloor \iota_k \rfloor)^{n_k}\big) \right\}_{1\le j\le \sharp(P,x)}$$

which concludes this case.

Similar arguments apply for all other syntactic constructs, so we get $\lfloor P \rfloor_\phi :: \lfloor \mathcal{I} \rfloor_P$, and $\lfloor \mathcal{I} \rfloor_P = \lfloor \mathcal{I} \rfloor$ by construction.

Proposition 4

Proof. Choose an arbitrary total order $\preceq$ over $\mathcal{L}$ and define, for each location $\ell \in \mathcal{L}(P)$:

$$\phi(\ell) = \begin{cases} \phi(c(\ell)) & \text{if } \ell \in \operatorname{dom} c \text{ and } \operatorname{pol} \ell = \uparrow \\ \sharp\{m \; ; \operatorname{subj} m = \operatorname{subj} \ell, m \preceq \ell\} & \text{otherwise} \end{cases}$$

Actions on a given name are numbered consecutively according to the chosen order over $\mathcal{L}$, so the numbering is clearly valid for all locations except the $\ell \in \operatorname{dom} c$ such that $\operatorname{pol} \ell = \uparrow$. For those locations, we use the fact that P is interfaced: in this case there are as many occurrences of $\operatorname{subj}_P \ell$ and $\operatorname{subj}_P c(\ell)$, and c establishes a bijection between them. As a consequence, ϕ is a numbering of P.

We first prove that c is a pairing of $\lfloor P \rfloor_\phi$. We have $\operatorname{pol}_{\lfloor P \rfloor_\phi} m = \operatorname{pol}_P m$ for all m, hence the polarity condition holds for all locations. Let $\ell \in \operatorname{dom} c$, then $\operatorname{subj}_P \ell$ is either hidden or action-bound in P. If $\operatorname{subj}_P \ell$ is hidden, then $\operatorname{subj}_P c(\ell) = \overline{\operatorname{subj}_P \ell}$, and by construction of $\lfloor P \rfloor_\phi$ we know that $\operatorname{subj}_{\lfloor P \rfloor_\phi} \ell$ is hidden, and by the first case of the definition ϕ we have $\phi(c(\ell)) = \phi(\ell)$, so $\operatorname{subj}_{\lfloor P \rfloor_\phi} c(\ell) = \overline{\operatorname{subj}_{\lfloor P \rfloor_\phi} \ell}$. If $\operatorname{subj}_P \ell$ is action-bound, then there is a location m and an integer i such that $\operatorname{subj}_P \ell = \operatorname{obj}_P^i m$ and $\operatorname{subj}_P c(\ell) = \operatorname{obj}_P^i c(m)$. By construction of ϕ we have $\phi(\ell) = \phi(c(\ell))$. Set $j = \phi(\ell) + \sum_{k<i} \sharp(P, \operatorname{obj}_P^k m)$, so that $\operatorname{subj}_{\lfloor P \rfloor_\phi} \ell = \operatorname{obj}^j_{\lfloor P \rfloor_\phi} m$. By the interfacing hypothesis we have $\sharp(P, \operatorname{obj}_P^k m) = \sharp(P, \operatorname{obj}_P^k c(m))$, hence we also have $\operatorname{subj}_{\lfloor P \rfloor_\phi} c(\ell) = \operatorname{obj}^j_{\lfloor P \rfloor_\phi} c(m)$. Therefore c is a valid pairing of $\lfloor P \rfloor_\phi$.

We now prove that c is maximal among pairings of $\lfloor P \rfloor_\phi$. For this, assume there is a location $\ell \in \mathcal{L}(\lfloor P \rfloor_\phi) \setminus \operatorname{dom} c$, and take ℓ minimal. Let $x = \operatorname{subj}_{\lfloor P \rfloor_\phi} \ell$. By proposition 3 $\lfloor P \rfloor_\phi$ is deterministic, so the multiplicity of x in the interface

derivation of $\lfloor P \rfloor_\phi$ is 1. If x is free, then ℓ cannot be in the domain of any pairing (be it greater than c or not). If x is hidden, then $\bar{x}$ has one occurrence at a location m, and m cannot be in the domain of c, otherwise we would have $c(m) = \ell$; therefore c can be extended with the pair (ℓ, m), which contradicts the maximality of c. Otherwise x is action-bound, so there is a location m and an integer i such that $\mathrm{obj}^i_{\lfloor P \rfloor_\phi}\, m = x$. If m was in the domain of c then c could be extended so that $c(\ell)$ is the location of the occurrence of $\mathrm{obj}^i_{\lfloor P \rfloor_\phi}\, c(m)$, which would contradict the maximality of c, so m is not in the domain of c, but this contradicts the minimality of ℓ. Hence c is maximal.

B Proofs about the type system

Proposition 5

Proof. Point 1 is straightforward. Point 2 is proved by induction on the typing derivation of P:

- Case $a{\multimap}\bar{b} \vdash a : A, \bar{b} : A^\perp$. Since $P = a{\multimap}\bar{b}$ we have $\mathcal{I} = a : 1\iota, \bar{b} : 1\bar{\iota}$ such that ι is positive. By construction $[\![\iota]\!]$ is a positive formula and $[\![\bar{\iota}]\!] = [\![\iota]\!]^\perp$.
- Case $\bar{a}(z_1 \dots z_k).P \vdash \Gamma, \bar{a} : \parr_{i=1}^k A_i$. By induction and nullary assignements, we have $P :: \mathcal{I}', \bar{a} : 0{\uparrow}(1\iota_1 \dots 1\iota_k), \{z_i : 1\iota_i\}_{i=1}^k$ The result follows by applying an action rule of interfacing.
- Case $a(z_1 \dots z_k).\prod_{i=1}^k P_i \vdash (\Gamma_i)_{i=1}^k, a : \bigotimes_{i=1}^k A_i$. By induction we have $\left\{P_i :: \mathcal{I}_i, z_i : 1\iota_i\right\}_{i=1}^k$ such that all elements are distinct. The result follows by applying a sequence of parallel rules of interfacing and one action rule.
- Case $(\boldsymbol{\nu} a)(P \mid Q) \vdash \Gamma, \Delta$. By induction we have $P :: \mathcal{I}, a : 1{\downarrow}\iota$ and $Q :: \mathcal{J}, \bar{a} : 1{\uparrow}\bar{\iota}$ such that $\mathcal{I} \cap \mathcal{J} = \emptyset$. The result follows by applying a parallel rule of interfacing and a rule $(\boldsymbol{\nu} a)$.

Proposition 6

Proof. We actually prove a more general statement: Let $(P_j)_{j \in J}$ be a finite family of a terms with pairwise disjoint location sets, let R be an equivalence over $\bigcup_{j \in J} \mathcal{L}(P_j)$. For each $i \in I$, assume a family of type derivations $(\Pi_{i,j} : P_j \vdash \Gamma_{i,j})_{j \in J}$ that globally respects R, that is such that for all $(\ell, m) \in R$, with $\ell \in \mathcal{L}(P_j)$ and $m \in \mathcal{L}(P_k)$, either $\mathrm{pol}_{P_j}\, \ell = \mathrm{pol}_{P_k}\, m$ and $\Pi_{i,j}(\ell) = \Pi_{i,k}(m)$, or $\mathrm{pol}_{P_j}\, \ell = \neg\, \mathrm{pol}_{P_k}\, m$ and $\Pi_{i,j}(\ell) = \Pi_{i,k}(m)^\perp$. Then there is a family of derivations $(\Pi_j : P_j \vdash \Gamma_j)_{j \in J}$ that globally respects R and a family of type substitutions $(\sigma_i)_{i \in I}$ such that for each i, j we have $\Gamma_{i,j} = \Gamma_j \sigma_i$. The statement of the proposition is the case where J is a singleton.

If I or J is empty, then the result holds trivially, so we can assume that they are not. We proceed by induction on the multiset of the sizes of the P_i.

- If each P_j is a forwarder $a_j{\multimap}\bar{b}_j$, then for each $i \in I$ we have $\Gamma_{i,j} = a_j : A_i^\perp, \bar{b}_j : A_i$ for some formula A_j. We can thus assume that $(a_j, \bar{b}_j) \in R$ for each $j \in J$. Let S be the equivalence relation over J such that $(j, k) \in S$

if $(a_j, a_k) \in R$ or $(a_j, \bar{b}_k) \in R$. By hypothesis, for each $i \in I$ the family $(\Pi_{i,j})_{j\in J}$ globally respects R, so for all $(j,k) \in S$ we have $A_j = A_k$. For each equivalence class A of J, pick a fresh type variable α_A. For each $j \in J$, define $\Gamma_j = a_j : \alpha_{\bar{j}}^{\perp}, \bar{b}_j : \alpha_{\bar{j}}$, and for each $i \in I$ define σ_i as the substitution that maps each $\alpha_{\bar{j}}$ to the formula A_j. Then the families $(\Gamma_j)_{j\in J}$ and $(\sigma_i)_{i\in I}$ are appropriate solutions.

– If some P_j has the shape $(\boldsymbol{\nu} a)P'$, then the last rule of any type derivation of P_j is a cut rule, so we can decompose P_j as $(\boldsymbol{\nu} a)(Q \mid Q')$. For each $i \in I$ we have a derivation

$$\frac{Q \vdash \Delta_i, a : A_i \qquad Q' \vdash \Delta'_i, \bar{a} : A_i^{\perp}}{(\boldsymbol{\nu} a)(Q \mid Q') \vdash \Delta_i, \Delta'_i}\ (\text{cut})$$

for some Δ_i, Δ'_i, A_i with $\Gamma_{i,j} = \Delta_i, \Delta'_i$. Since a and $\bar{a}$ must have opposite types, we can assume that the equivalence R contains the pair (a, a'). Then we can apply the induction hypothesis to the family obtained by replacing each $\Pi_{i,j} : P_j \vdash \Gamma_{i,j}$ by the derivations of $Q \vdash \Delta_i, a : A_i$ and $Q' \vdash \Delta'_i, \bar{a} : A_i^{\perp}$, under the same constraint R. This way, we get two types Δ and Δ', a formula A and a family of substitutions σ_i such that $\Delta_i = \Delta\sigma_i$, $\Delta'_i = \Delta'\sigma_i$ and $A_i = A\sigma_i$, and derivations of $Q \vdash \Delta, a : A$ and $Q' \vdash \Delta', \bar{a} : A^{\perp}$. The fact that the types a and $\bar{a}$ match is guaranteed by the constraint in R. We conclude using the same family of transitions, replacing the derivations for Q and Q' by a derivation of $P_j \vdash \Delta, \Delta'$.

– If some P_j has the shape $a(z_1 \dots z_n).P'$, then the last rule of any type derivation is a n-ary tensor rule, so we can decompose P as $a(\boldsymbol{z}).\prod_{k=1}^{n} Q_k$, and for each $i \in I$ we have a derivation

$$\frac{\left\{Q_k \vdash \Delta_k, z_k : A_{i,k}\right\}_{k=1}^{n}}{a(\boldsymbol{z}).\prod_{k=1}^{n} Q_k \vdash (\Delta_{i,k})_{k=1}^{n}, a : \bigotimes_{k=1}^{n} A_{i,k}}\ (\otimes)$$

We can conclude by induction hypothesis on the family obtained by replacing, for each $i \in I$, the derivation $\Pi_{i,j} : P_j \vdash \Gamma_i$ by the derivations of the $Q_k \vdash \Delta_k, z_k : A_{i,k}$.

– If some P_j has the shape $\bar{a}(\boldsymbol{z}).P'$, we conclude as above using a par rule.

No other case may occur, because of the typability constraint, since the family of type derivations is supposed to be non-empty.

Lemma 3. *For all typed term $(\boldsymbol{\nu} a)(\bar{a}^{\ell}(\zeta).P \mid a^{m} \multimap \bar{b}^{n}) \vdash \Gamma, \bar{b} : A$ we have the typing $\bar{b}^{n}(\zeta).P \vdash \Gamma, \bar{b} : A$*

Lemma 4. *Let $\Pi : P \vdash \Gamma$ be a type derivation and let $P \to_{ex}^{c} P'$ be an execution step. Then there is a type derivation $\Pi_0 : C[R] \vdash \Gamma$, where $C[\,]$ is a context made of parallel compositions and hidings, such that $\Pi \to_{sc}^{\emptyset} \Pi_0$, $P \equiv C[R]$ and the*

typing of R has the form

$$\dfrac{\bar{a}_0(\zeta).S \vdash \Gamma, \bar{a}_0 : A^\perp \quad \pi_1}{\dfrac{\vdots \qquad\qquad \pi_n}{\dfrac{[\cdots] \vdash \Gamma, \bar{a}_n : A^\perp \qquad a_n(\bar{\zeta}).T \vdash \Delta, a_n : A}{R \vdash \Gamma, \Delta}\ (\text{cut})}\ (\text{cut})}\ (\text{cut})$$

for some $n \geq 0$, where π_i is the derivation of $a_{i-1} \multimap \bar{a}_i \vdash a_{i-1} : A, \bar{a}_i : A^\perp$ by an axiom rule, and with

$$R = (\boldsymbol{\nu} a_n)(\cdots(\boldsymbol{\nu} a_1)((\boldsymbol{\nu} a_0)(\bar{a}_0(\zeta).S \mid a_0 \multimap \bar{a}_1) \mid a_1 \multimap \bar{a}_2) \cdots \mid a_n(\bar{\zeta}).T).$$

Proof. Consider the set of all type derivations that can be reached from the typing of P by commutation between cuts and commutativity steps. For contradiction, assume that this set does not contain a derivation for a term $C[R]$ of the appropriate form. Choose a derivation Π that contains a sub-tree Π_1 of the form

$$\dfrac{\bar{a}_0(\zeta).S \vdash \Gamma, \bar{a}_0 : A^\perp \quad \pi_1}{\dfrac{\vdots \qquad\qquad \pi_k}{[\cdots] \vdash \Gamma, \bar{a}_k : A^\perp}\ (\text{cut})}\ (\text{cut})$$

for a maximal k. The derivation Π must contain an introduction rule Π_2 for $a_k : A$. Moreover, the sub-tree above and this introduction must be sub-trees of the premisses of a cut Π_3 between $\bar{a}_k : A^\perp$ and $a_k : A$. The considered execution step of P involves a_k and $\bar{a}_k$, so these names cannot be prefixed by actions, hence the corresponding sub-trees Π_1 and Π_2 can only be above cut rules in Π. By commutation steps, we can thus bring Π_1 and Π_2 as premisses of the cut rule Π_3, which contradicts the maximality of k.

Theorem 1

Proof. By lemma 3, there is a type derivation $\Pi_0 : C[R] \vdash \Theta$ where R has the shape of the statement of the lemma. Call Π_R the sub-tree of Π_0 that types R. On Π_R we can apply logical cut elimination steps to eliminate the axioms $\pi_1, \ldots, \pi_n$, which leads to

$$\Pi_R \to_{sc}^{(\bar{a}_0, a_0)} \cdots \to_{sc}^{(\bar{a}_{n-1}, a_{n-1})} \dfrac{\bar{a}_n(\zeta).S \vdash \Theta, \bar{a}_n : A^\perp \quad a_n(\bar{\zeta}).T \vdash \Delta, a_n : A}{(\boldsymbol{\nu} a_n)(\bar{a}_n(\zeta).S \mid a_n(\bar{\zeta}).T) \vdash \Theta, \Delta}\ (\text{cut})$$

The premisses of the last cut rule are introduction rules for the formulas A and $A^\perp$. Hence the formula A can be decomposed as $\bigotimes_{i=1}^{k} A_i$ and the premisses are

$$\dfrac{S \vdash \Theta, \{z_i : A_i^\perp\}_{i=1}^{k}}{\bar{a}_n(z_1 \ldots z_k).S \vdash \Theta, \bar{a}_n : A^\perp}\ (\parr) \qquad \dfrac{\{T_i \vdash \Delta_i, \bar{z}_i : A_i\}_{i=1}^{k}}{a_n(\bar{z}_1 \ldots \bar{z}_k).T \vdash \Delta, a_n : A}\ (\otimes)$$

with $\Delta = \Delta_1, \ldots, \Delta_k$ and $T = T_1 \mid \cdots \mid T_k$. By a logical cut elimination step, annotated $(\bar{a}_n, a_n)$, we get a derivation of

$$R' = (\boldsymbol{\nu} z_1)(\cdots(\boldsymbol{\nu} z_k)(S \mid T_k)\cdots \mid T_1) \vdash \Theta, \Delta$$

We can easily check that the reduction $R \to^c_{ex} R'$ holds, up to structural congruence, hence $P \equiv C[R] \to^c_{ex} C[R']$. By proposition 9, this implies $C[R'] \equiv P'$.

C Proofs on the wireless proof net and reductions

Proposition 13. *Let Π a* $\mathrm{MLL_u}$ *sequent calculus proof. If we write $\widetilde{R}$ the transformation of cut wires of a proof net R in a cut-node set of a wireless proof net obtained by erasing, then we have $\widetilde{[\Pi]} = [\![\Pi]\!]$ i.e. wireless proof nets are proof nets without wires. If $R = [\![\Pi]\!]$ then there exists C a subset of the complement graph of R such that $R \cup C = [\Pi]$.*

As consequence of the contractility property, a paired graph is connected (i.e. as usual, the undirected underlying graph is connected) if and only if it rewrites to a single node (possibly with loops) by the previous rewriting system. A paired graph is acyclic if and only if it rewrites without loops by the previous rewriting system.

Proposition 14. *Proof nets and wireless proof nets with empty cut-node set are contractile.*

Proof. Easy by induction on proof nets and wireless proof net.

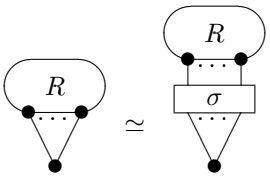
Examples of cut-choice. If there is $A \in \mathscr{C}_R$ such that for all cut-paths ρ of label A, the paired graph $R \cup \rho$ is cyclic, then there is no cut-choice in R because of the contractibility condition required.

Another similar example is when we consider for a given $B \in \mathscr{C}_R$, the complete graph K_B whose nodes are in $\mathscr{C}_B$. If there is $A \in \mathscr{C}_R$ such that for all cut-paths ρ of label A, the paired graph $R \cup (\cup_{B \in \mathscr{C}_R \setminus A} K_B) \cup \rho$ is disconnected, then there is no cut-choice in R because of the contractibility condition required.

Definitions 15 and 20 imply what follows:

Lemma 5. *Let Π a type derivation such that $[\![\Pi]\!] \Rightarrow_{C_A} R'$, then there is a sequence of cut elimination steps such that $\Pi \to^c_{sc} \Pi'$ and $R' = [\![\Pi']\!]$.*

Example of permutation equivalence. Let R be a wireless proof net with $\{n_i\}_{i=1}^k$ conclusion nodes labelled respectively A_i. Let $R \cup v$ be a wireless proof net extending R with a tensor node v whose incoming edges are from $\{n_i\}_{i=1}^k$. Each permutation σ of the total order over the incoming edges of v preserving the label $F = \otimes_{i=1}^k A_i$ of v gives an equivalent wireless proof net. E.g. if for all $i \in \{1, \ldots, k\}$ we have $A_i = A$ then permutations of $\{1, \ldots, k\}$ give equivalent wireless proof nets.


R
≃
R
σ